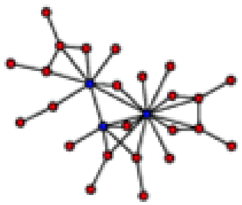
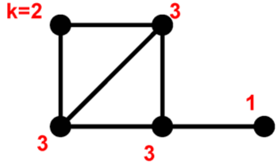


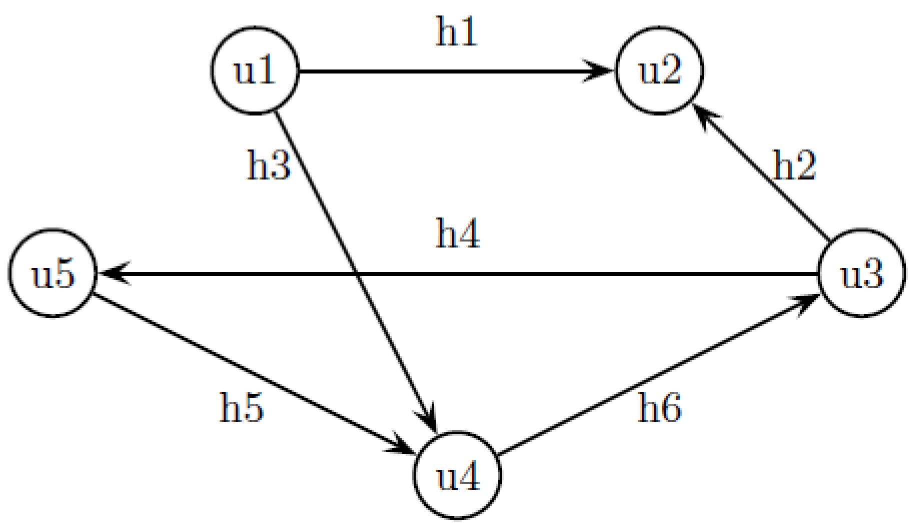
Data Analysis I

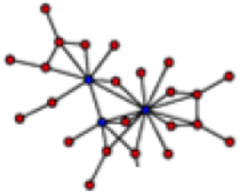
Graph Representation

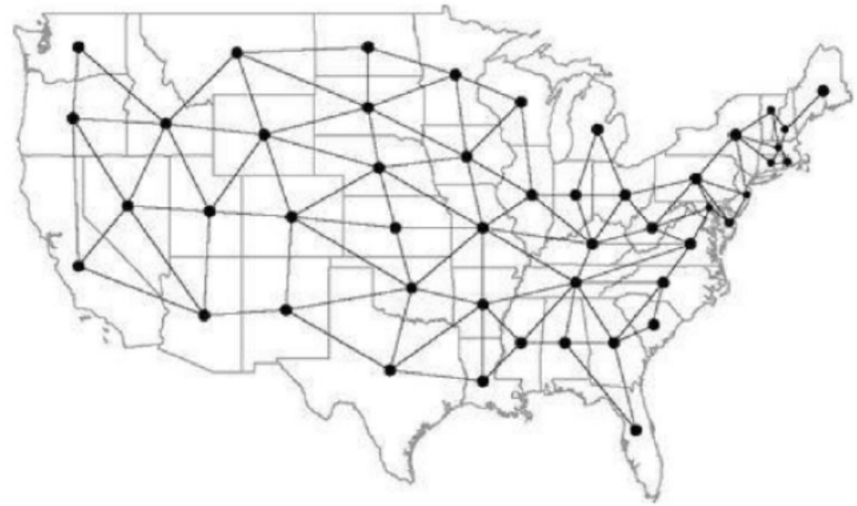


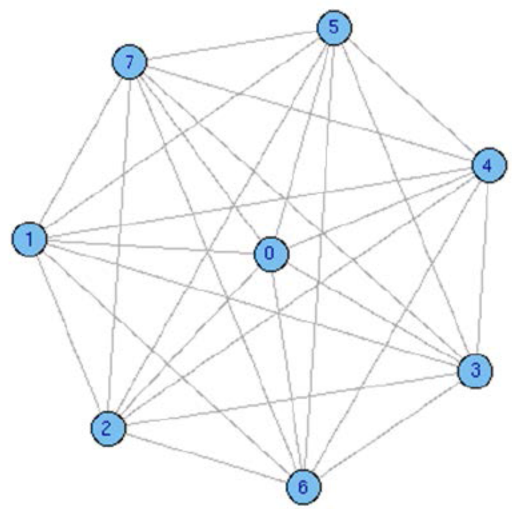
Graph representation

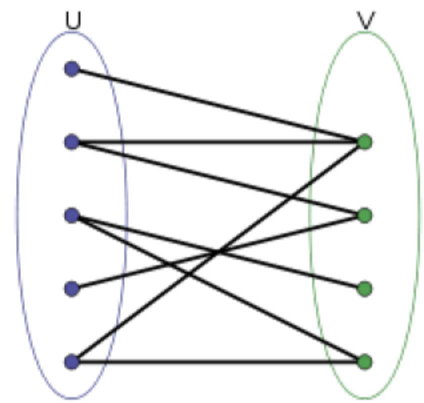
- By image (drawing) ☺

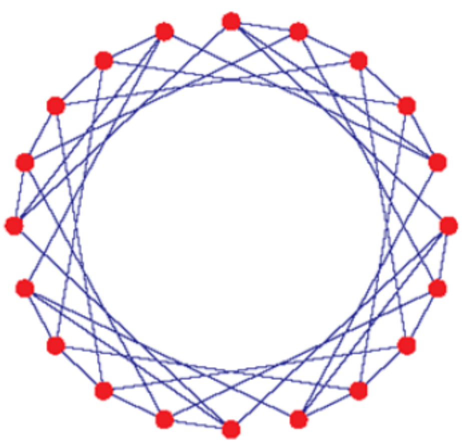


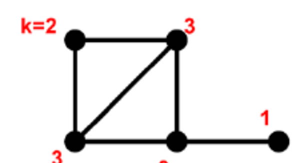


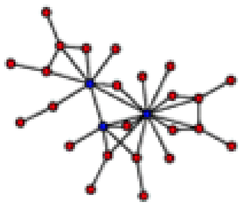



5
7
4
1
0
3
2
6


U
V

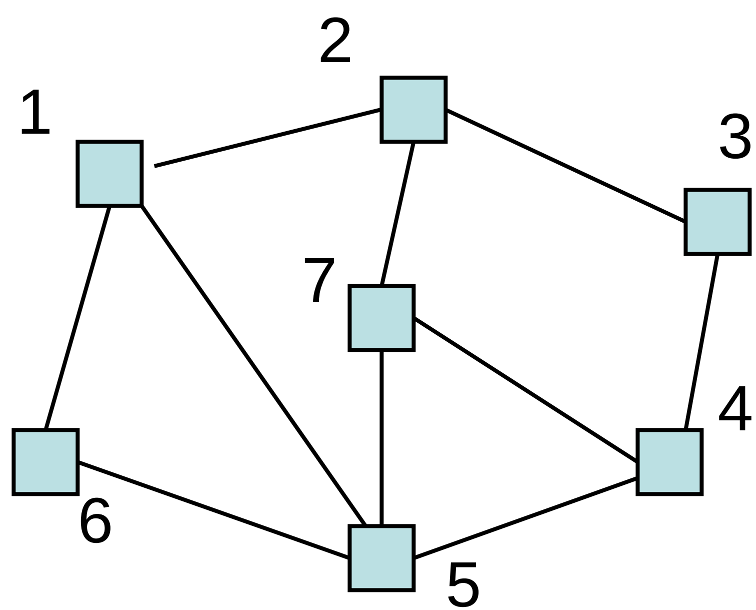


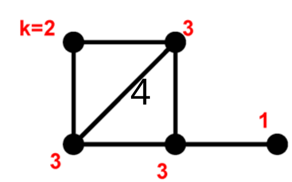

k=2
3
1
3
3

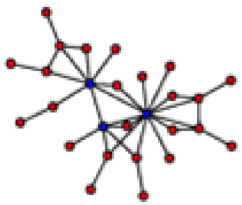


Implementing a Graph

- To program a graph data structure, what information would we need to store?
  - For each vertex?
  - For each edge?

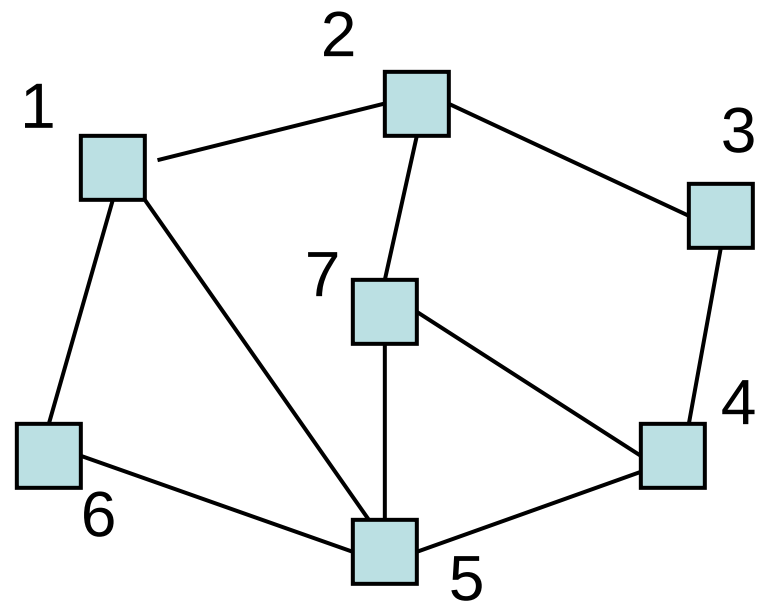


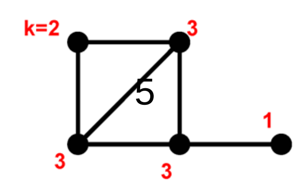


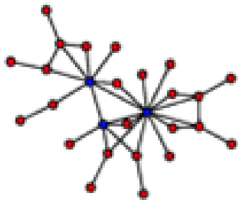


Implementing a Graph

- What kinds of questions would we want to be able to answer (quickly?) about a graph *G*?
  - Where is vertex *v*?
  - Which vertices are adjacent to vertex *v*?
  - What edges touch vertex *v*?
  - What are the edges of *G*?
  - What are the vertices of *G*?
  - What is the degree of vertex *v*?

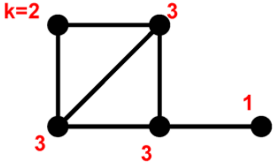






Representation

- There are different ways to represent a graph
  - List of edges
  - List of lists: Node list and list of neighbors
  - Adjacency matrix



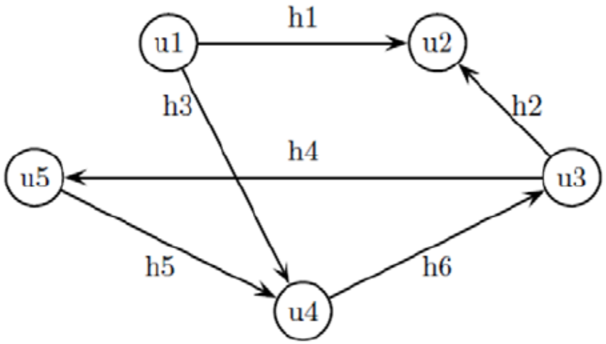
Adjacency matrix

$$A_{ij} = \begin{cases} w_{ij} & \text{if } i \text{ and } j \text{ are connected} \\ 0 & \text{otherwise} \end{cases}$$

If A represents an *unweighted network*, then $w_{ij} = 1$ for all i, j.

For undirected graph (omit a direction)

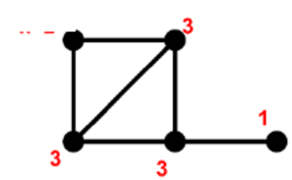
$$A = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

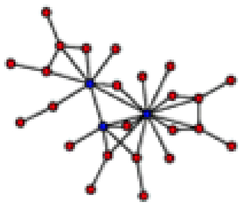


For directed graph

$$A = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

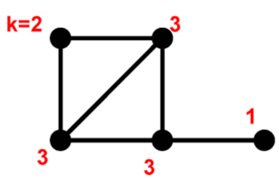
https://en.wikipedia.org/wiki/Adjacency_matrix





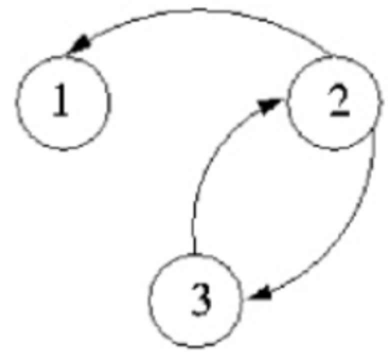
Adjacency matrix

- Adjacency is chosen on the ordering of vertices. Hence, there as are as many as n! such matrices.
- The adjacency matrix of undirected graphs are symmetric ($a_{ij} = a_{ji}$) (why?) → redundant information for undirected graphs
- When there are relatively few edges in the graph the adjacency matrix is a **sparse matrix**
- Directed multigraphs can be represented by using a_{ij} = number of edges from v_i to v_j , undirected similarly



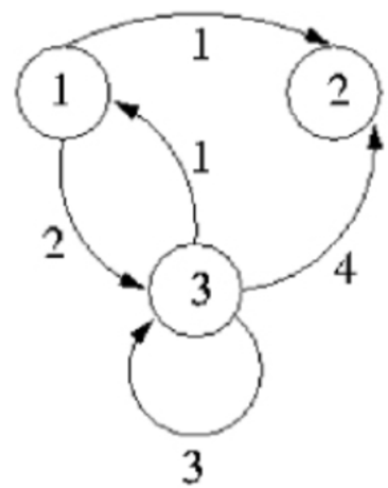
Modifications of adjacency matrix

- Weighted adjacency matrix
- Distance matrix (e.g. result of Floyd's algorithm (later))
- Similarity matrix (later)



Directed

$$A = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$



Directed Weighted

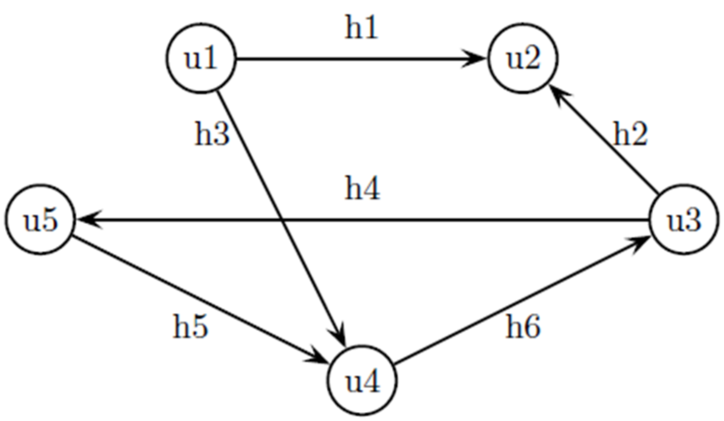
$$A = \begin{pmatrix} 0 & 1 & 2 \\ 0 & 0 & 0 \\ 1 & 4 & 3 \end{pmatrix}$$

Incidence matrix

- Let G = (V, E) be an unditected graph. Then the incidence matrix with respect to this ordering of V and E is the n x m matrix $B=[b_{ij}]$, where

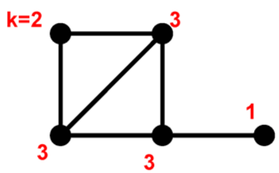
$$b_{ij} = \begin{cases} 1 \text{ when edge } e_j \text{ is incident with } v_i \\ 0 \text{ otherwise} \end{cases}$$

$$B = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$



$$B = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & -1 \\ 0 & 0 & -1 & 0 & -1 & 1 \\ 0 & 0 & 0 & -1 & 1 & 0 \end{bmatrix}$$

https://en.wikipedia.org/wiki/Incidence_matrix



List of lists: Node list and list of neighbors (Adjacency list)

https://en.wikipedia.org/wiki/Adjacency_list

Undirected graph

$v_1 \rightarrow v_2, v_4$

$v_2 \rightarrow v_1, v_3$

$v_3 \rightarrow v_2, v_4, v_5$

$v_4 \rightarrow v_1, v_3, v_5$

$v_5 \rightarrow v_3, v_4$

Directed graph

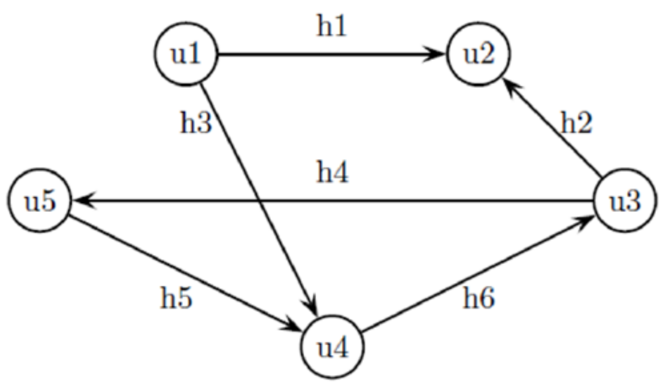
$v_1 \rightarrow v_2, v_4$

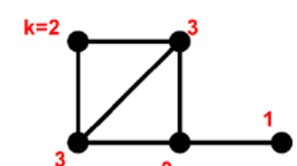
$v_2 \rightarrow$

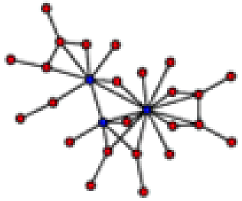
$v_3 \rightarrow v_2, v_5$

$v_4 \rightarrow v_3$

$v_5 \rightarrow v_4$

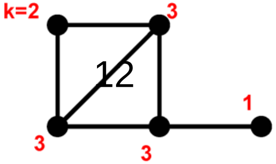


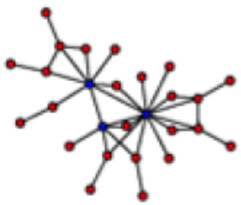




Node list and list of neighbors: Pros and Cons

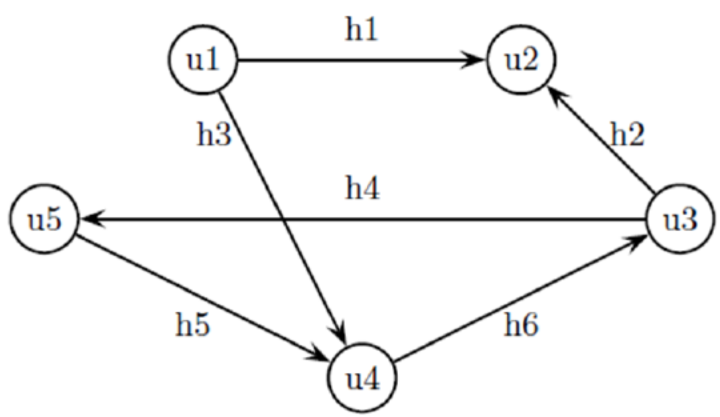
- Adjacency list stores edges as individual linked lists of references to each vertex's neighbors
- *advantages:*
  - new nodes can be added easily
  - new nodes can be connected with existing nodes easily
  - "who are my neighbors" easily answered
- *disadvantages:*
  - determining whether an edge exists between two nodes: O(average degree)

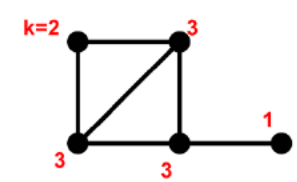




List of edges

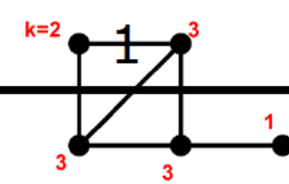
Undirected graph	Directed graph
(v_1, v_2)	(v_1, v_2)
(v_1, v_4)	(v_1, v_4)
(v_2, v_3)	(v_3, v_2)
(v_3, v_4)	(v_3, v_5)
(v_3, v_5)	(v_4, v_3)
(v_4, v_5)	(v5, v4)

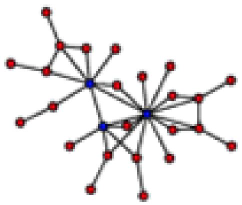




Runtime table

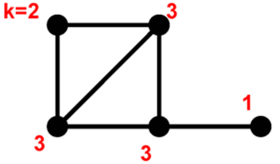
■ n vertices, m edges ■ no parallel edges ■ no self-loops	Edge List	Adjacency List	Adjacency Matrix
Space	$n + m$	$n + m$	n^2
Finding all adjacent vertices to v	m	deg(v)	n
Determining if v is adjacent to w	m	deg(v)	1
adding a vertex	1	1	n^2
adding an edge to v	1	1	1
removing vertex v	m	deg(v)	n^2
removing an edge from v	m	deg(v)	1

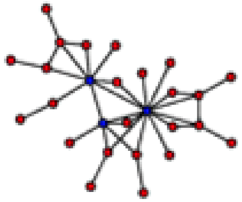




Data Analysis I

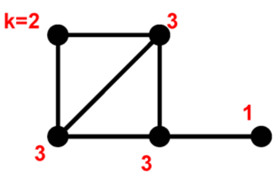
Measures and Metrics





References

- Newman, M. (2010). *Networks: an introduction*. Oxford University Press. [168-193]
- Zaki, M. J., Meira Jr, W. (2014). *Data Mining and Analysis: Fundamental Concepts and Algorithms*. Cambridge University Press. [97-102]
- Aaron Clauset. *Network Analysis and Modeling* http://tuvalu.santafe.edu/~aaronc/courses/5352/csci5352_2017_L2.pdf
- Albert-László Barabási. Network Science http://barabasi.com/networksciencebook/ Chapter 2



Structural (topological) properties

- **Degree**
- Local - degree *d* of a node v_i (as the number of its neighbors)

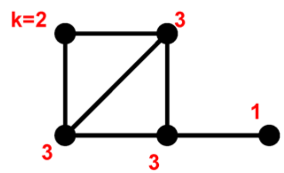
$$d_i = \sum_j \mathrm{A}(i, j)$$

- Global – mean (average) degree $\mu_d = \frac{\sum_i d_i}{n}$

- indegree / outdegree (by taking the summation over the incoming / outgoing edges) as follows:

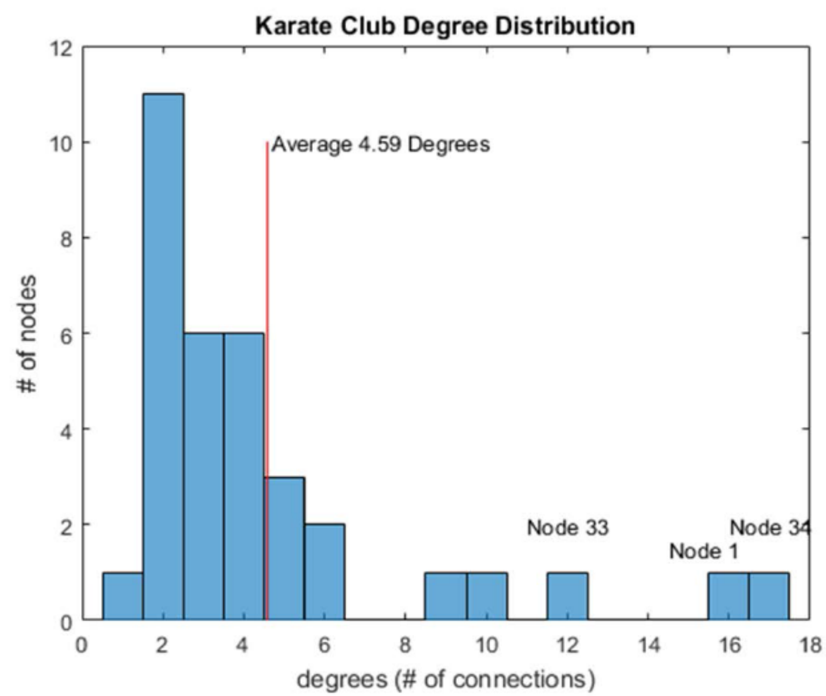
$$id(v_i) = \sum_j \mathrm{A}(j, i)$$

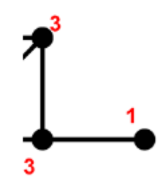
$$od(v_i) = \sum_j \mathrm{A}(i, j)$$



Degree centrality

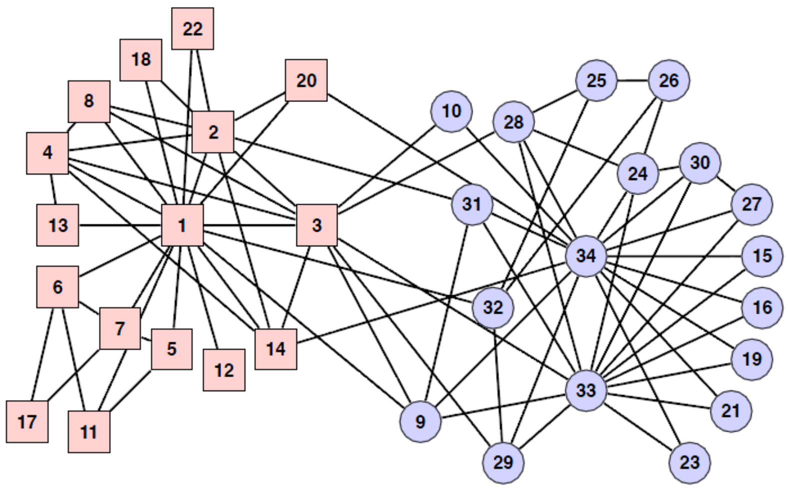
- Degree centrality
- Social network - the higher the degree, e.g., a more influential person
- The citation network - the higher the degree (more references), the "greater the impact of the publication on scientific research„
- Zachary’s carate club

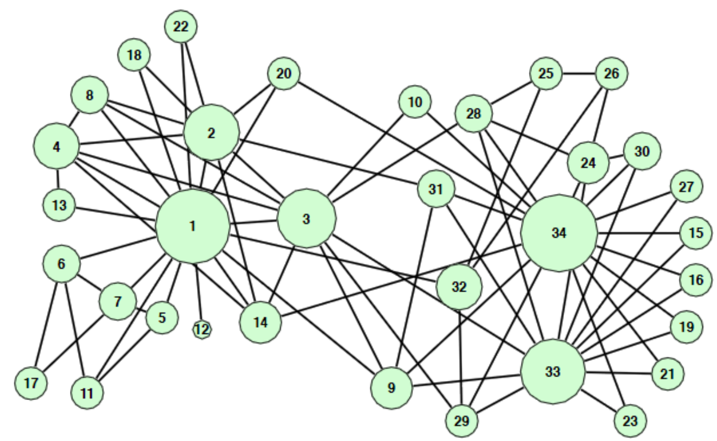




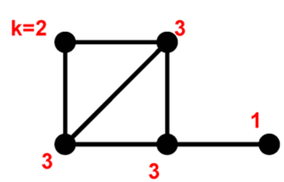
Degree centrality

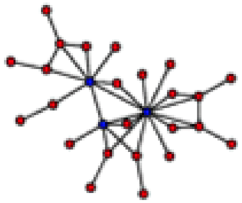
- Zachary's carate club





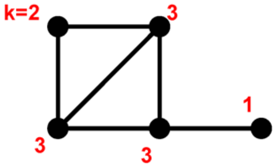
group 1	1	2	3	4	5	6	7	8	11	12	13	14	17	18	20	22		
k	16	9	10	6	3	4	4	4	3	1	2	5	2	2	3	2		
k/m	0.10	0.06	0.06	0.04	0.02	0.03	0.03	0.03	0.02	0.01	0.01	0.03	0.01	0.01	0.02	0.01		
group 2	9	10	15	16	19	21	23	24	25	26	27	28	29	30	31	32	33	34
k	5	2	2	2	2	2	2	5	3	3	2	4	3	4	4	6	12	17
k/m	0.03	0.01	0.01	0.01	0.01	0.01	0.01	0.03	0.02	0.02	0.01	0.03	0.02	0.03	0.03	0.04	0.08	0.11





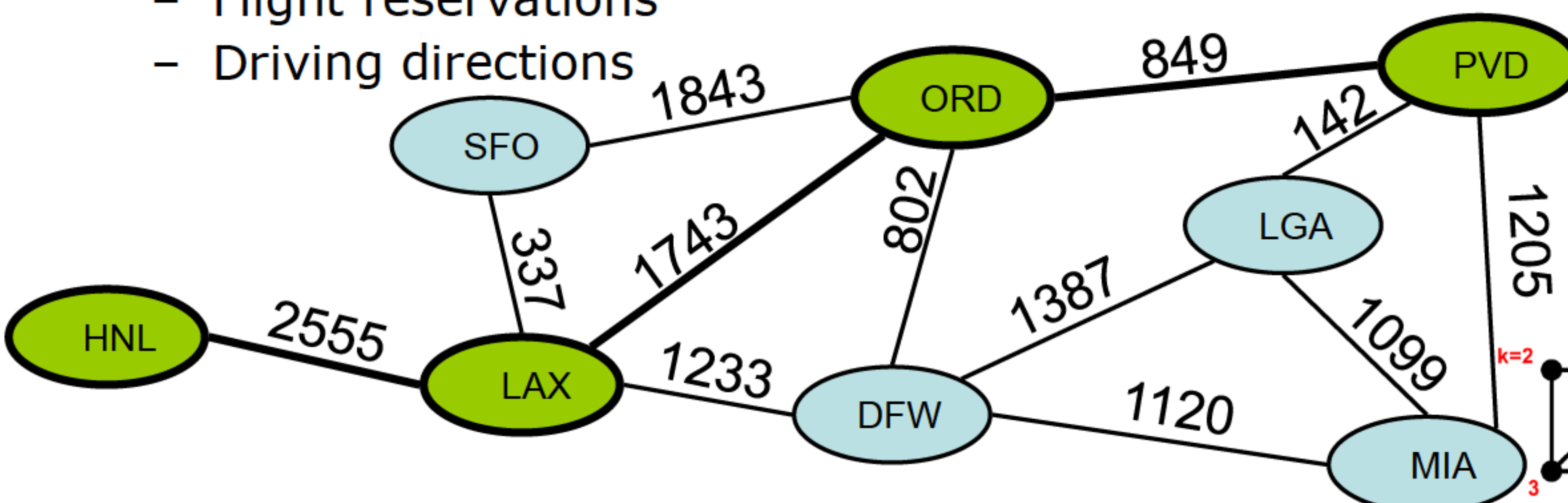
Path

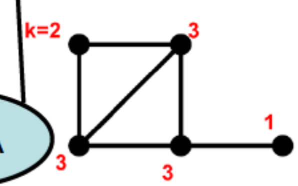
- A path in a graph is a finite or infinite sequence of edges which joins a sequence of vertices which, by most definitions, are all distinct (and since the vertices are distinct, so are the edges).
- The shortest path problem is the problem of finding a path between two vertices (or nodes) in a graph such that the sum of the weights of its constituent edges is minimized.
- The distance between two vertices in a graph is the number of edges in a shortest path (also called a graph geodesic) connecting them. This is also known as the geodesic distance.

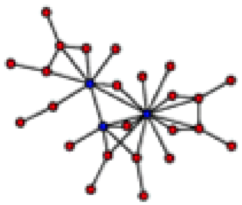


Shortest Path Problem

- Given a weighted graph and two vertices ***u*** and ***v***, we want to find a path of **minimum** total weight between ***u*** and ***v.***
  - Length of a path is the sum of the weights of its edges.
  - In unweighted graph the weights od edges are = 1.
- Example:
  - Shortest path between Providence and Honolulu
- Applications
  - Internet packet routing
  - Flight reservations
  - Driving directions







Structural properties

- **Path**
- Local
  - The eccentricity *e(v_i)* of a node v_i is the maximum distance from v_i to any other node in the graph

$$e(v_i) = \max_j \left\{ d(v_i, v_j) \right\}$$

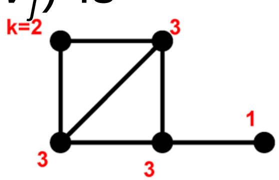
- Global
  - The diameter *d(G)*, is the maximum eccentricity of any vertex in the graph

$$d(\mathrm{G}) = \max_i \left\{ e(v_i) \right\} = \max_{i,j} \left\{ d(v_i, v_j) \right\}$$

  - Mean distance (Average Path Length) of a connected graph is given as

$$\mu_{\mathrm{L}} = \frac{\sum_i \sum_{j>i} d(v_i, v_j)}{\binom{n}{2}} = \frac{2}{n(n-1)} \sum_i \sum_{j>i} d(v_i, v_j)$$

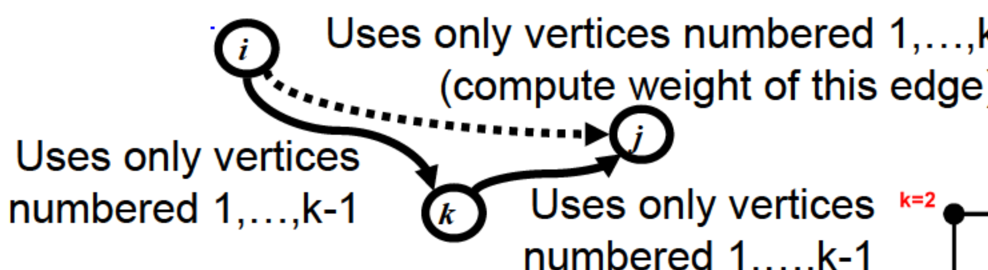
  - where *n* is the number of nodes in the graph, and *d(v_i ,v_j)* is the distance between v_i and v_j .

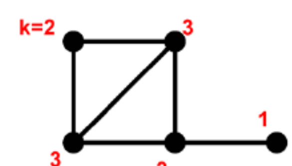


All-Pairs Shortest Paths

- Find the distance between every pair of vertices in a weighted directed graph G.
- Floyd-Warshall algorithm
- https://en.wikipedia.org/wiki/Floyd%E2%80%93Warshall_algorithm
- https://www.youtube.com/watch?v=4OQeCuLYj-4

```
Algorithm AllPair(G) {assumes vertices 1,…,n}
 for all vertex pairs (i,j)
   if i = j
      C_0[i,i] ← 0
   else if (i,j) is an edge in G
      C_0[i,j] ← weight of edge (i,j)
   else
      C_0[i,j] ← + ∞
 for k ← 1 to n do
   for i ← 1 to n do
     for j ← 1 to n do
        C_k[i,j] ← min{C_{k-1}[i,j], C_{k-1}[i,k]+C_{k-1}[k,j]}
 return C_n
```





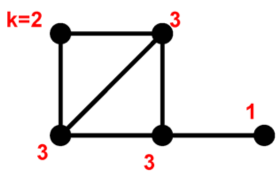
Example: Floyd-Warshall

$$v_1 \longrightarrow (v_2, 3), (v_3, 5)$$
$$v_2 \longrightarrow (v_3, 1), (v_4, 2), (v_5, 4)$$
$$v_3 \longrightarrow$$
$$v_4 \longrightarrow (v_3, 9), (v_5, 1)$$
$$v_5 \longrightarrow (v_1, 2), (v_3, 8)$$

$$C^0 = \left[\begin{array}{c|cccc} 0 & 3 & 5 & x & x \\ \hline x & 0 & 1 & 2 & x \\ x & x & 0 & x & x \\ x & x & 9 & 0 & 1 \\ 2 & x & 8 & x & 0 \end{array}\right] \quad C^1 = \left[\begin{array}{c|c|ccc} 0 & 3 & 5 & x & x \\ \hline x & 0 & 1 & 2 & x \\ \hline x & x & 0 & x & x \\ x & x & 9 & 0 & 1 \\ 2 & 5 & 7 & x & 0 \end{array}\right]$$

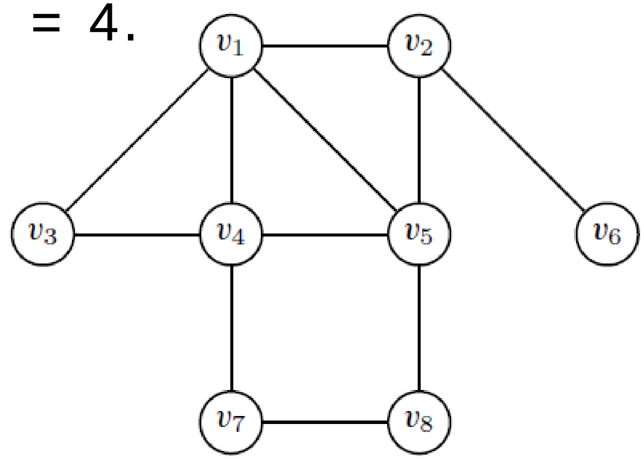
$$C^2 = \left[\begin{array}{cc|c|cc} 0 & 3 & 4 & 5 & x \\ x & 0 & 1 & 2 & x \\ \hline x & x & 0 & x & x \\ \hline x & x & 9 & 0 & 1 \\ 2 & 5 & 6 & 7 & 0 \end{array}\right] \quad C^3 = \left[\begin{array}{ccc|c|c} 0 & 3 & 4 & 5 & x \\ x & 0 & 1 & 2 & x \\ x & x & 0 & x & x \\ \hline x & x & 9 & 0 & 1 \\ \hline 2 & 5 & 6 & 7 & 0 \end{array}\right]$$

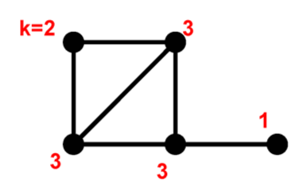
$$C^4 = \left[\begin{array}{cccc|c} 0 & 3 & 4 & 5 & 6 \\ x & 0 & 1 & 2 & 3 \\ x & x & 0 & x & x \\ x & x & 9 & 0 & 1 \\ \hline 2 & 5 & 6 & 7 & 0 \end{array}\right] \quad C^5 = \left[\begin{array}{ccccc} 0 & 3 & 4 & 5 & 6 \\ 5 & 0 & 1 & 2 & 3 \\ x & x & 0 & x & x \\ 3 & 6 & 7 & 0 & 1 \\ 2 & 5 & 6 & 7 & 0 \end{array}\right]$$

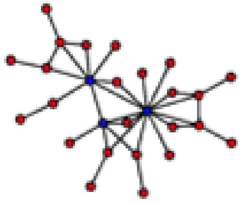


Structural properties

- Degree sequence of the graph on the figure is (4,4,4,3,2,2,2,1) and therefore its degree frequency distribution is given as $(N_0,N_1,N_2,N_3,N_4) = (0,1,3,1,3)$
- Mean degree is 2.75
- The degree distribution is given as $f(k) = P(X = k) = N_k/n$
  $(f(0), f(1), f(2), f(3), f(4)) = (0,0.125,0.375,0.125,0.375)$
- The eccentricity of node v_4 is 3, because the node farthest from it is v_6 and $d(v4,v6) = 3$.
- The diameter of the graph is $d(G) = 4$, as the largest distance over all the pairs is $d(v6,v7) = 4$.

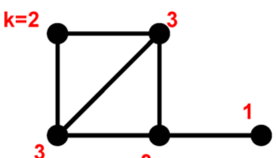


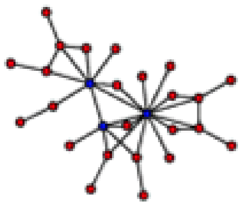




References

- Newman, M. (2010). *Networks: an introduction*. Oxford University Press. [168-193]
- Zaki, M. J., Meira Jr, W. (2014). *Data Mining and Analysis: Fundamental Concepts and Algorithms*. Cambridge University Press. [97-102]
- Aaron Clauset. *Network Analysis and Modeling* http://tuvalu.santafe.edu/~aaronc/courses/5352/csci5352_2017_L2.pdf
- Albert-László Barabási. Network Science http://barabasi.com/networksciencebook/ Chapter 2





Structural properties (global)

- **Network dimension** – networks are usually large, i.e. the number of vertices is large
- Network density - Networks are usually sparse
  - Sparse vs. dense graph (net)
  - The resolution is often vague, depending on the context
  - Most often
    - Sparse graph - $n \approx km$, $m = O(n)$
    - Dense graph – $m = \Theta(n^2)$
  - **Density** H:
    - mean degree $\langle d \rangle = 2m/n$ → after the of division by highest degree (n-1) →
    - $2m/n*(n-1)$ →
    - $H = m / 0.5*n*(n-1)$, $H \in \langle 0,1 \rangle$